University of British Columbia
CPSC 111, Intro to Computation
2009W2: Jan-Apr 2010

Tamara Munzner

More Class Design II

Lecture 31, Wed Mar 31 2010

borrowing from slides by Kurt Eiselt

http://www.cs.ubc.ca/~tmm/courses/111-10

News

- A3 out today
    - due Fri Apr 16 5pm

News: Midterm Correction Lab

- you can earn **up to** 10% of marks that you missed back by working through what you got wrong to find correct answers

- do your new version on separate sheets of paper
  - don't mark up the original midterm

Bunny Class Warmup

Question 4: **[15 marks]**
Now let's use Java to simulate bunnies! (Why? Because everybody likes bunnies!) In our simulation, each bunny is on a grid at some location defined by an X-coordinate and a Y-coordinate. Also, each bunny has some number of energy units measured in carrot sticks. (X-coordinates, Y-coordinates, and the number of carrot sticks are integer values.) Bunnies can hop north, south, east, or west. When a bunny hops to the north, the bunny's Y-coordinate is increased by 1, and the X-coordinate remains unchanged. When a bunny hops to the west, the bunny's X-coordinate is decreased by 1, and the Y-coordinate remains unchanged. Same idea for hops east (X-coordinate increased by 1, Y-coordinate unchanged) and south (Y-coordinate decreased by 1, X-coordinate unchanged). Note that making one hop requires a bunny to eat one carrot stick, and when a bunny has eaten all of his or her carrot sticks, that bunny can't hop.

Use Java to create a Bunny class which can be used to generate Bunny objects that behave as described above. When a new Bunny object is created, the Bunny always starts at coordinates X = 10, Y = 10, and the Bunny has 5 carrot sticks. Your Bunny class definition must include a hop(int direction) method, and a displayInfo() method. The direction parameter is 12 for north, 3 for east, 6 for south, and 9 for west (like a clock face). The hop() method should test to make sure that the Bunny has not eaten all the carrot sticks – if the Bunny still has carrot sticks, the hop() method should update coordinates as explained above and print the message "hop". If no carrot sticks remain, it should just print the message "This bunny can't hop".
The displayInfo() method should print the Bunny's location and number of remaining carrot sticks. Below is a simple test program that could be used to test your Bunny class definition, followed by the output we'd expect to see when using this test program with your Bunny class definition.

```java
public class BunnyTest
{
  public static void main(String[] args)
  {
    System.out.println("Testing Peter");
    Bunny peter = new Bunny();
    peter.displayInfo();
    peter.hop(12);
    peter.hop(12);
    peter.hop(9);
    peter.displayInfo();
    System.out.println("Testing Emily");
    Bunny emily = new Bunny();
    emily.displayInfo();
    emily.hop(9);
    emily.hop(9);
    emily.hop(9);
    emily.hop(12);
    emily.hop(9);
    emily.hop12();
    emily.displayInfo();
  }
}
```

```
> java BunnyTest
Testing Peter
This bunny is at position 10,10
This bunny has 5 carrot sticks remaining
hop
hop
hop
This bunny is at position 9,12
This bunny has 2 carrot sticks remaining
Testing Emily
This bunny is at position 10,10
This bunny has 5 carrot sticks remaining
hop
hop
hop
hop
hop
This bunny can't hop
This bunny is at position 6,11
This bunny has 0 carrot sticks remaining
>
```

More Bunnies

How could we keep track of a herd of bunnies?

We could make an array of bunnies.

More Bunnies

```
public class BunnyTest1
{
  public static void main (String[] args)
  {
    Bunny[] myBunnyHerd = new Bunny[10];

    myBunnyHerd[0] = new Bunny(3,6,4,"Foofoo");
    myBunnyHerd[1] = new Bunny(7,4,2,"Peter");
    myBunnyHerd[3] = new Bunny(9,2,3,"Ed");

    for(int i = 0; i < myBunnyHerd.length; i++)
    {
      if (myBunnyHerd[i] != null)
      {
        myBunnyHerd[i].hop(3);
        System.out.println(myBunnyHerd[i]);
      }
    }
  }
}
```

Even More Bunnies

Question 5: **[16 marks]**
The world desperately needs better bunny management software, so please help by writing a BunnyHerd class. A BunnyHerd object holds an array of Bunny objects. Your BunnyHerd class definition should include the following four methods:

constructor Expects two parameters, an integer representing the maximum number of bunnies in the herd, and a String for the name of the herd.

`addBunny(int xPos, int yPos, int carrots,String name)` Expects four parameters, the X- and Y-coordinates of the bunny, the number of carrots, and the name. This method creates a new Bunny object and stores the reference to the object in the next available location in the BunnyHerd object.

`deleteBunny(String name)` Expects one parameter, the name of the bunny. This method removes from the BunnyHerd object all references to bunnies with the given `name` by overwriting those references with the `null` pointer. This method does not change the pointer to the next available location in the BunnyHerd object.

`printHerd()` This method uses the `toString()` method of the Bunny object to print information about every Bunny in the herd.

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
    int number = 4;
    System.out.println("main: number is " + number);
    method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
    int number = 4;
    System.out.println("main: number is " + number);
    method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
    System.out.println("main: number is " + number);
    method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
    method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
7   System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
7   System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?

Parameter Passing

Consider the following program:

```
 public class ParamTest1
 {
   public static void main (String[] args)
   {
 1   int number = 4;
 2   System.out.println("main: number is " + number);
 3   method1(number);
 7   System.out.println("main: number is now " + number);
   }

   public static void method1(int x)
   {
 4   System.out.println("method1: x is " + x);
 5   x = x * x;
 6   System.out.println("method1: x is now " + x);
   }
 }
```

What's printed?

```
main: number is 4
```

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
7   System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?

```
main: number is 4
method1: x is 4
```

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
7   System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?

```
main: number is 4
method1: x is 4
method1: x is now 16
```

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
7   System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?

```
main: number is 4
method1: x is 4
method1: x is now 16
?????????????????????
```

Parameter Passing

Consider the following program:

```
  public class ParamTest1
  {
    public static void main (String[] args)
    {
1     int number = 4;
2     System.out.println("main: number is " + number);
3     method1(number);
7     System.out.println("main: number is now " + number);
    }

    public static void method1(int x)
    {
4     System.out.println("method1: x is " + x);
5     x = x * x;
6     System.out.println("method1: x is now " + x);
    }
  }
```

What's printed?

```
main: number is 4
method1: x is 4
method1: x is now 16
main: number is now 4
```

Parameter Passing

Consider the following program:

```
  public class ParamTest1
  {
    public static void main (String[] args)
    {
1     int number = 4;
2     System.out.println("main: number is " + number);
3     method1(number);
7     System.out.println("main: number is now " + number);
    }

    public static void method1(int x)
    {
4     System.out.println("method1: x is " + x);
5     x = x * x;
6     System.out.println("method1: x is now " + x);
    }
  }
```

Why not 16?

```
main: number is 4
method1: x is 4
method1: x is now 16
main: number is now 4
```

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
7   System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

Because when the value in the `int` variable `number` is passed to `method1`,

Parameter Passing

Consider the following program:

```
public class ParamTest1
{
  public static void main (String[] args)
  {
1   int number = 4;
2   System.out.println("main: number is " + number);
3   method1(number);
7   System.out.println("main: number is now " + number);
  }

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

Because when the value in the `int` variable `number` is passed to `method1`, what really happens is that a copy of the value (4) in `number` is assigned to the parameter `x`.

Parameter Passing

Consider the following program:

```
  public class ParamTest1
  {
    public static void main (String[] args)
    {
1     int number = 4;
2     System.out.println("main: number is " + number);
3     method1(number);
7     System.out.println("main: number is now " + number);
    }

    public static void method1(int x)
    {
4     System.out.println("method1: x is " + x);
5     x = x * x;
6     System.out.println("method1: x is now " + x);
    }
  }
```

Because when the value in the `int` variable `number` is passed to `method1`, what really happens is that a copy of the value (4) in `number` is assigned to the parameter `x`. It's the value in `x` that's being modified here -- a copy of the value in `number`. The original value in `number` is not affected.

Parameter Passing

Will this program behave differently? Why or why not?

```
public class ParamTest2
{
  public static void main (String[] args)
  {
    int number = 4;
    System.out.println("main: number is " + number);
    method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int number)
  {
    System.out.println("method1: number is " + number);
    number = number * number;
    System.out.println("method1: number is now " + number);
  }
}
```

What's printed?

Parameter Passing

Will this program behave differently? Why or why not?

```
public class ParamTest2
{
  public static void main (String[] args)
  {
    int number = 4;
    System.out.println("main: number is " + number);
    method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int number)
  {
    System.out.println("method1: number is " + number);
    number = number * number;
    System.out.println("method1: number is now " + number);
  }
}
```

What's printed?

```
main: number is 4
method1: number is 4
method1: number is now 16
?????????????????????????
```

Parameter Passing

Will this program behave differently? Why or why not?

```
public class ParamTest2
{
  public static void main (String[] args)
  {
    int number = 4;
    System.out.println("main: number is " + number);
    method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int number)
  {
    System.out.println("method1: number is " + number);
    number = number * number;
    System.out.println("method1: number is now " + number);
  }
}
```

What's printed?

```
main: number is 4
method1: number is 4
method1: number is now 16
main: number is now 4
```

Parameter Passing

Will this program behave differently? Why or why not?

```
public class ParamTest2
{
  public static void main (String[] args)
  {
    int number = 4;
    System.out.println("main: number is " + number);
    method1(number);
    System.out.println("main: number is now " + number);
  }

  public static void method1(int number)
  {
    System.out.println("method1: number is " + number);
    number = number * number;
    System.out.println("method1: number is now " + number);
  }
}
```

Remember that a parameter declared in a method header has local scope, just like a variable declared within that method. As far as Java is concerned, `number` inside of `method1` is unrelated to `number` outside of `method1`. They are not the same variable.

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

What's printed?

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

```
main: foo is now: 4
```

What's printed?

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
```

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
```

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
?????????????????????
```

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
main: foo is now: 16
```

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

Why not 4?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
main: foo is now: 16
```

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

What's in `foo`? Is it the `int[]` array object?

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

What's in `foo`? Is it the `int[]` array object? No, it's the reference, or pointer, to the object.

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

What's in `foo`? Is it the `int[]` array object? No, it's the reference, or pointer, to the object. A copy of that reference is passed to `method1` and assigned to `x`.

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

What's in `foo`? Is it the `int[]` array object? No, it's the reference, or pointer, to the object. A copy of that reference is passed to `method1` and assigned to `x`. The reference in `foo` and the reference in `x` both point to the same object.

Parameter Passing

Now consider this program.

```
public class Ptest
{
  public static void main(String[] args)
  {
    int[] foo = new int[1];
    foo[0] = 4;
    System.out.println("main: foo is now: " + foo[0]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  public static void method1(int[] x)
  {
    System.out.println("method1: x is now: " + x[0]);
    x[0] = x[0] * x[0];
    System.out.println("method1: x is now: " + x[0]);
  }
}
```

When the object pointed at by `x` is updated, it's the same as updating the object pointed at by `foo`. We changed the object that was pointed at by both `x` and `foo`.

Parameter Passing

- Passing primitive types (int, double, boolean) as parameter in Java
  - "pass by value"
  - value in variable is copied
  - copy is passed to method
  - modifying copy of value inside called method has no effect on original value outside called method
    - modifying aka mutating

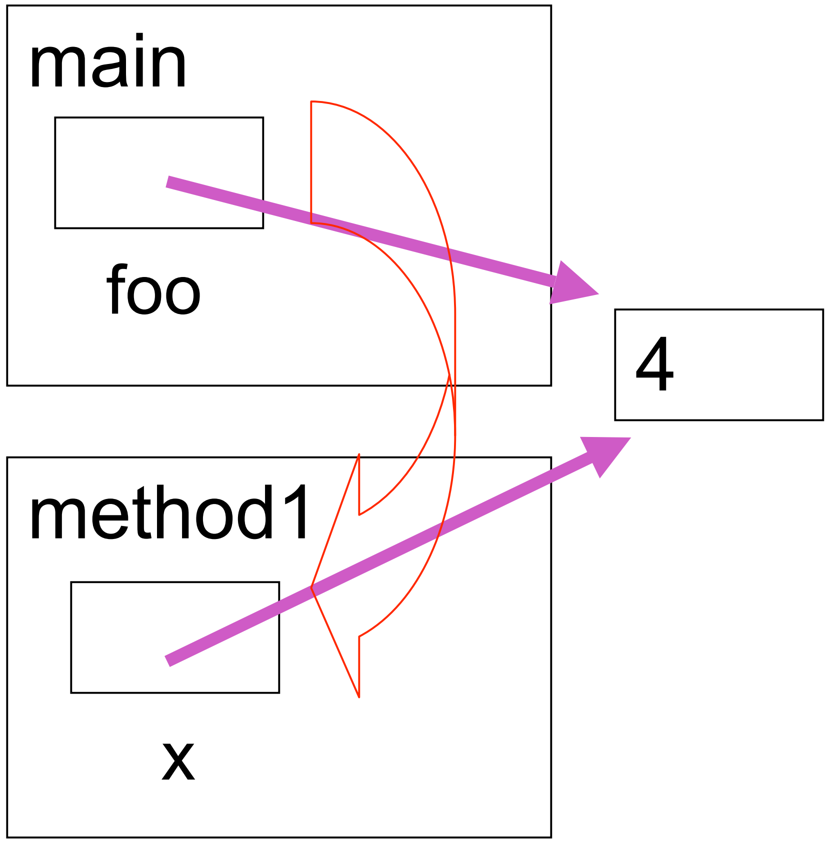
Parameter Passing

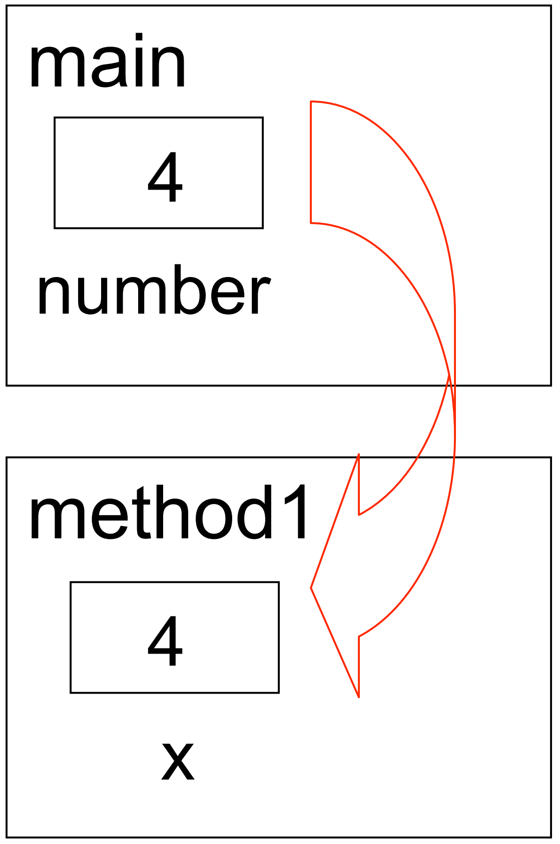
- Passing object as parameter in Java
  - "pass by reference"
  - objects could be huge, so do not pass copies around
  - pass copy of the object reference
    - object reference aka pointer
  - modifying object pointed to by reference inside calling method **does** affect object pointed to by reference outside calling method
    - both references point to **same object**

Parameter Passing Pictures

object as parameter:
copy of pointer made

prim as parameter:
copy of value

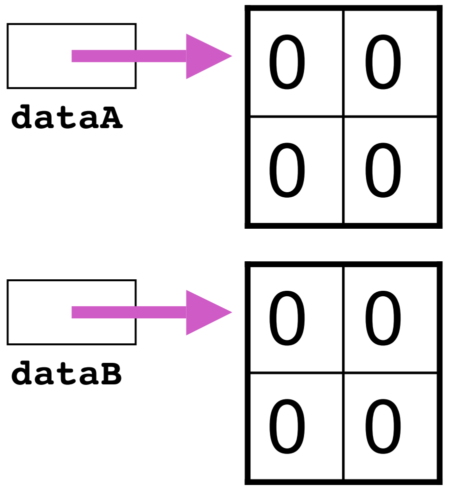




Midterm Q4 from 04W2

```
    int[][] dataA = { { 0, 0 }, { 0, 0 } };
    int[][] dataB = { { 0, 0 }, { 0, 0 } };
    process( dataA, dataB );
```

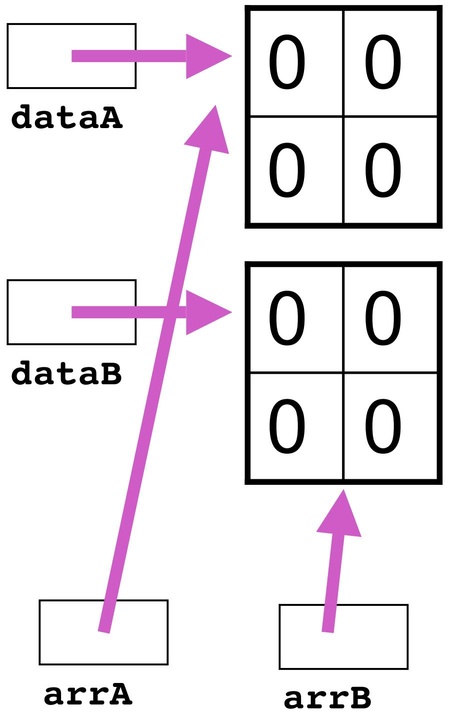
```
    public void process( int[][] arrA, int[][] arrB )
    {
        int row;
        int col;
        int[][] arrC = { { 1, 1, 1 }, { 1, 1, 1 } };
        arrA = arrC;
        for( row = 0; row < arrB.length; row++ )
        {
            for( col = 0; col < arrB[ row ].length; col++
    )
            {
                arrB[ row ][ col ] = row + col;
            }
        }
    }
```



Midterm Q4 from 04W2

```
    int[][] dataA = { { 0, 0 }, { 0, 0 } };
    int[][] dataB = { { 0, 0 }, { 0, 0 } };
    process( dataA, dataB );
```

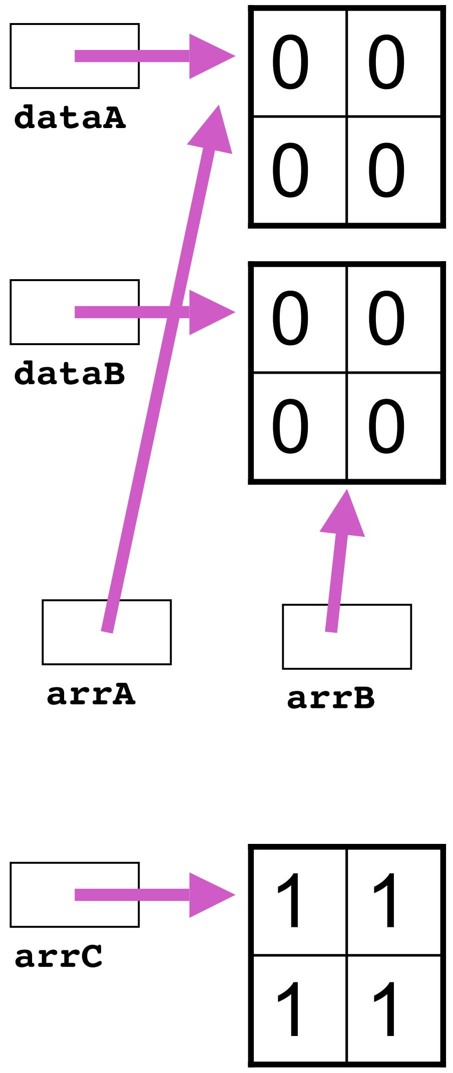
```
    public void process( int[][] arrA, int[][] arrB )
    {
       int row;
       int col;
       int[][] arrC = { { 1, 1, 1 }, { 1, 1, 1 } };
       arrA = arrC;
       for( row = 0; row < arrB.length; row++ )
       {
          for( col = 0; col < arrB[ row ].length; col++
    )
          {
             arrB[ row ][ col ] = row + col;
          }
       }
    }
```



Midterm Q4 from 04W2

```
    int[][] dataA = { { 0, 0 }, { 0, 0 } };
    int[][] dataB = { { 0, 0 }, { 0, 0 } };
    process( dataA, dataB );
```

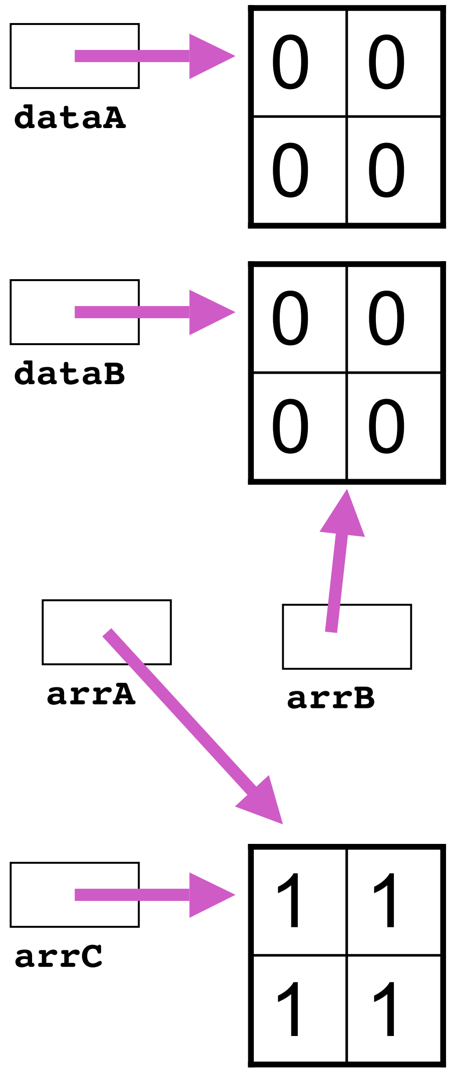
```
    public void process( int[][] arrA, int[][] arrB )
    {
        int row;
        int col;
        int[][] arrC = { { 1, 1, 1 }, { 1, 1, 1 } };
        arrA = arrC;
        for( row = 0; row < arrB.length; row++ )
        {
            for( col = 0; col < arrB[ row ].length; col++
        )
            {
                arrB[ row ][ col ] = row + col;
            }
        }
    }
```



Midterm Q4 from 04W2

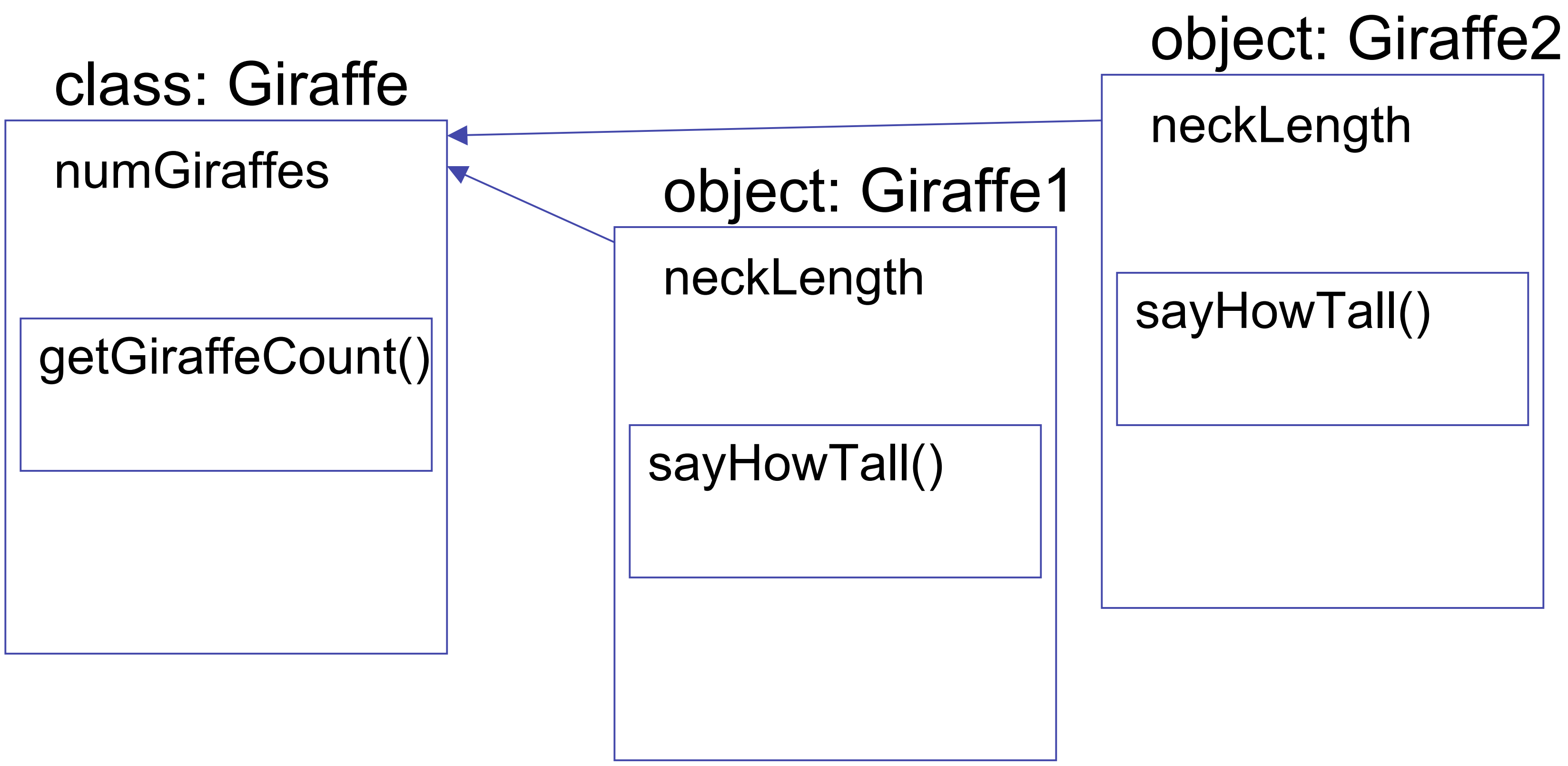
```
int[][] dataA = { { 0, 0 }, { 0, 0 } };
int[][] dataB = { { 0, 0 }, { 0, 0 } };
process( dataA, dataB );
```

```
public void process( int[][] arrA, int[][] arrB )
{
    int row;
    int col;
    int[][] arrC = { { 1, 1, 1 }, { 1, 1, 1 } };
    arrA = arrC;
    for( row = 0; row < arrB.length; row++ )
    {
        for( col = 0; col < arrB[ row ].length; col++
)
        {
            arrB[ row ][ col ] = row + col;
        }
    }
}
```



Review: Static Fields/Methods

- Static fields belong to whole class
  - nonstatic fields belong to instantiated object
- Static methods can only use static fields
  - nonstatic methods can use either nonstatic or static fields



Review: Variable Scope

- Scope of a variable (or constant) is that part of a program in which value of that variable can be accessed

Variable Scope

```
public class CokeMachine4
{
  private int numberOfCans;

  public CokeMachine4()
  {
    numberOfCans = 2;
    System.out.println("Adding another machine to your empire");
  }

  public int getNumberOfCans()
  {
    return numberOfCans;
  }

  public void reloadMachine(int loadedCans)
  {
    numberOfCans = loadedCans;
  }
```

- numberOfCans variable declared inside class but not inside particular method
    - scope is entire class: can be accessed from anywhere in class

Variable Scope

```
public class CokeMachine4
{
  private int numberOfCans;

  public CokeMachine4()
  {
    numberOfCans = 2;
    System.out.println("Adding another machine to your empire");
  }

  public double getVolumeOfCoke()
  {
    double totalLitres = numberOfCans * 0.355;
    return totalLitres;
  }

  public void reloadMachine(int loadedCans)
  {
    numberOfCans = loadedCans;
  }
```

- totalLitres declared within a method
  - scope is method: can only be accessed from within method
  - variable is local data: has local scope

Variable Scope

```
public class CokeMachine4
{
  private int numberOfCans;

  public CokeMachine4()
  {
    numberOfCans = 2;
    System.out.println("Adding another machine to your empire");
  }

  public int getNumberOfCans()
  {
    return numberOfCans;
  }

  public void reloadMachine(int loadedCans)
  {
    numberOfCans = loadedCans;
  }
```

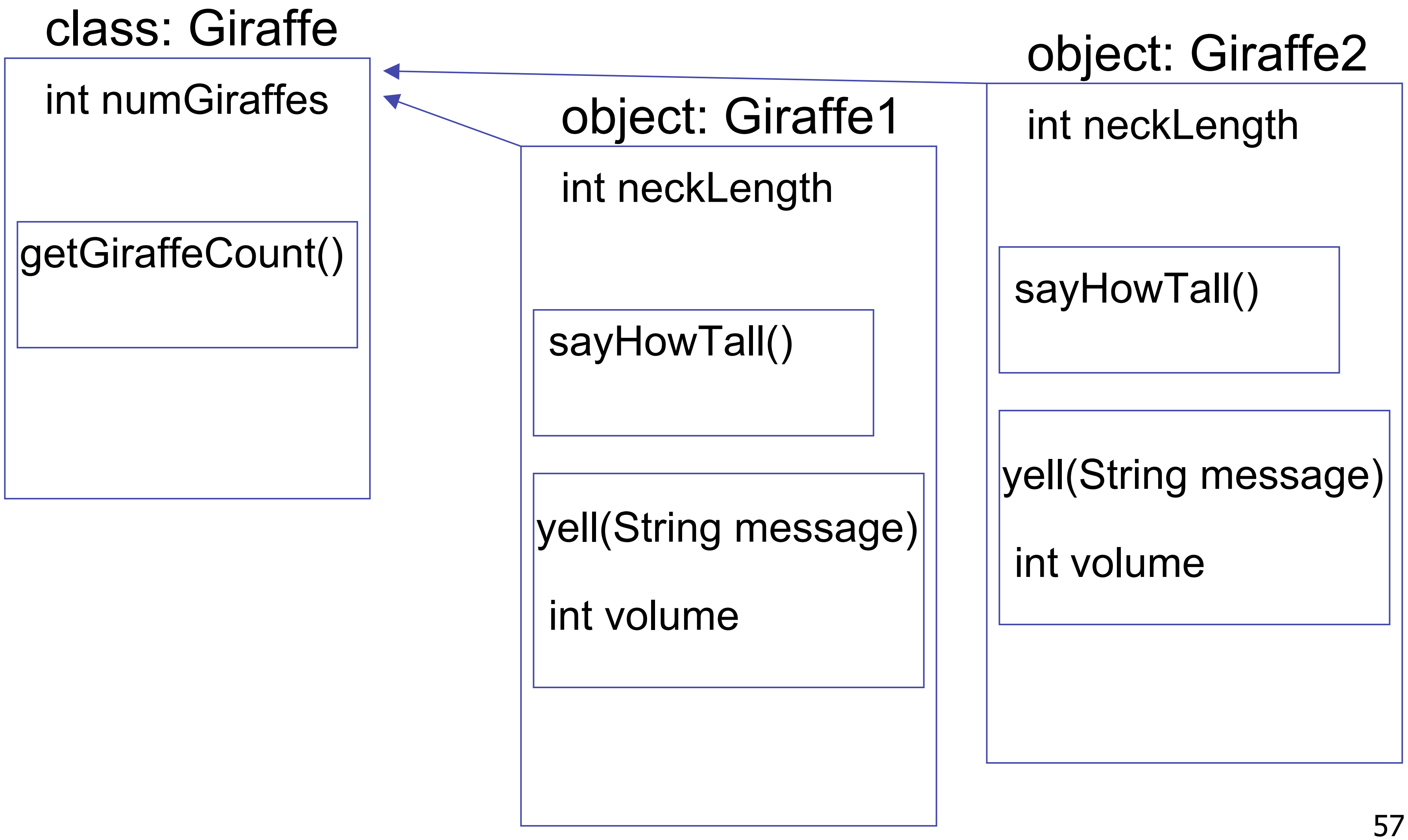
- loadedCans is method parameter
  - scope is method: also local scope
  - just like variable declared within parameter
  - accessed only within that method

Variable Types

- Static variables
  - declared within class
  - associated with class, not instance
- Instance variables
  - declared within class
  - associated with instance
  - accessible throughout object, lifetime of object
- Local variables
  - declared within method
  - accessible throughout method, lifetime of method
- Parameters
  - declared in parameter list of method
  - acessible throughout method, lifetime of method

Variable Types

- Static? Instance? Local? Parameters?



Questions?